Graph Based Pattern Recognition

Exercise 5

Basis

- Chapters 8 and 9

Submission

- The submission takes place online on ILIAS.
- Solutions to the theory tasks must be submitted as `*.pdf` file. Other formats will not be accepted.
- Source code for the implementation tasks must be submitted as `*.py` files. Source code that cannot be executed will not be accepted.
- Individual submissions or submissions in teams of two are allowed (hand in only one copy per group). In the source code file, include the *names and matriculation numbers* of both group members in the first two lines as comments.

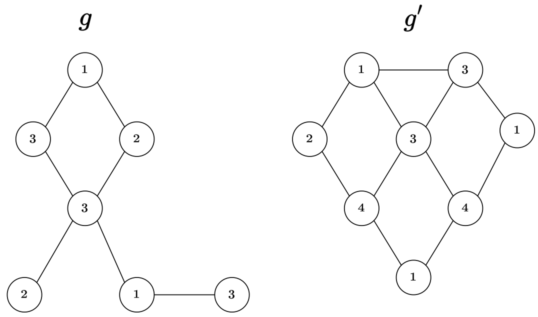
Dates

- Briefing: 26.04.2023
- Submission: 10.05.2023
- Debriefing: 10.05.2023

Theoretical Tasks

1. Given the following graphs g and g', compute the angle $\measuredangle(g, g')$ between them by means of a *node feature kernel*.

   A node feature kernel is a type of kernel function that computes the graph similarity based on a their *feature vectors*. A feature vector is a vector of numerical values that describes the characteristics of the nodes. Each element in the vector corresponds to a specific feature or attribute of the node, such as the number of occurrences of one specific label.

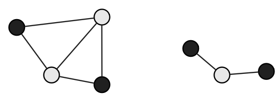


2. Discuss the major benefits of the kernel trick and kernel machines for graph-based pattern recognition.

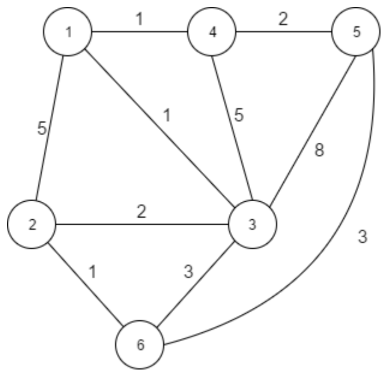
3. Given the following dissimilarity matrix $\mathbf{D}$, compute the kernel matrix of the von Neumann diffusion kernel $\mathbf{K}$ (with $\lambda = 0.1$). Determine the value of t at which the sum of the diffusion kernel matrices converges (the step t where the difference between two consecutive matrices in the infinite sum is less than or equal to $\epsilon = 10^{-3}$ (i.e., $||\mathbf{M}_t - \mathbf{M}_{t-1}||_2 \leq \epsilon$)).

$$\mathbf{D} = \begin{bmatrix} 0 & 5 & 8 & 9 & 6 \\ 5 & 0 & 3 & 7 & 7 \\ 8 & 3 & 0 & 4 & 6 \\ 9 & 7 & 4 & 0 & 1 \\ 6 & 7 & 6 & 1 & 0 \end{bmatrix}$$

4. Compute and illustrate the direct product graph for the following two graphs and the adjacency matrices $\mathbf{A}_\times^n$ with $n = \{1, 2, 3\}$ (the nodes are labeled with a binary label 'black' or 'gray'). Illustrate the meaning of an entry $a_{ij} = 4$ in $\mathbf{A}_\times^2$, and an entry $a_{ij} = 8$ in $\mathbf{A}_\times^3$.



5. Apply the Floyd Transformation to the following graph.



Implementation Tasks

The practical exercise consists of two tasks: Implementing the shortest-path kernel and an enumerating graph kernel. First, go to the ILIAS's webpage of the course and download/unzip `Exercise_5.zip` in your `PR_Lecture` folder.

1. In this first task, your goal is to implement the shortest-path kernel presented in the lecture notes (Section 9.2) (use equation 9.5 in the lecture notes for the definition of $\kappa_{\text{path}}(e_1, e_2)$). Navigate to `PR_lecture/Exercise_5/ex5_a.py` and complete the missing part of the source code.

   Once it is implemented, compute the shortest-path kernel between all pairs of graphs in `Exercise_5/graphs`. To this end, create an $N \times N$ matrix $\mathbf{K} = (k_{ij})$, where $N =$ is the number of graphs in the `PR_lecture/Exercise_5/graphs` directory. Matrix $\mathbf{K}$ contains the results of your shortest-path kernel implementation for all pairs of graphs. More formally, entry k_{ij} at position i, j corresponds to the shortest-path kernel between graph g_i and graph g_j.

   Save your matrix as `SPK_results.csv` in the result folder.

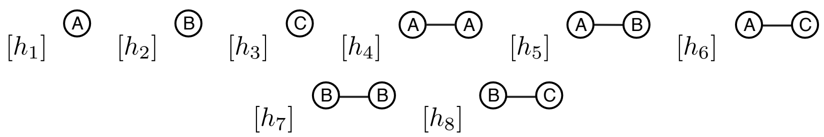
2. Some graph kernels are defined based on explicit enumerations of predefined substructures (cycles, trees, subgraphs, walks, etc.). Let us assume a reference set of substructures is given $H = \{h_1, \dots, h_n\}$. One can then define a mapping by means of

   $$\varphi(g) = (f(h_1, g), \dots, f(h_n, g))$$

   where $f(h_i, g)$ counts the frequency of h_i in g. A possible graph kernel is then given by

   $$\kappa(g, g') = \langle \varphi(g), \varphi(g') \rangle$$

   Implement the graph kernel $\kappa(g.g')$ by using the following reference graphlets h_1 to h_8.



   Once it is implemented, compute the kernel between all pairs of graphs in `Exercise_5/graphs`. To this end, create an $N \times N$ matrix $\mathbf{K} = (k_{ij})$, where $N =$ is the number of graphs in the `PR_lecture/Exercise_5/graphs` directory. Matrix $\mathbf{K}$ contains the results of your enumerating kernel implementation for all pairs of graphs. More formally, entry k_{ij} at position i, j corresponds to the enumerating kernel between graph g_i and graph g_j.

   Save your matrix as `EK_results.csv` in the result folder.

Submission

For the coding part, you must submit a `.zip` file containing the following files. Additionally, include your solution for the theoretical tasks as `*.pdf` in the same `.zip` file.

```
Exercise_5
├── drawings
├── ex5_a.py
├── ex5_b.py
├── graphlets
│   ├── graph_00.graphml
│   ├── graph_01.graphml
│   ├── graph_02.graphml
│   ├── graph_03.graphml
│   ├── graph_04.graphml
│   ├── graph_05.graphml
│   ├── graph_06.graphml
│   └── graph_07.graphml
├── graphs
│   ├── graph_00.graphml
│   ├── graph_01.graphml
│   ├── graph_02.graphml
│   ├── graph_03.graphml
│   └── graph_04.graphml
├── __init__.py
├── results
│   ├── EK_results.csv
│   └── SPK_results.csv
└── utils.py
```